Proving termination by policy iteration

Damien Massé

Proving termination by policy iteration

Damien Massé[1]

Lab-Sticc, UMR 6285
UBO - Université Européenne de Bretagne
Brest, France

Abstract

In this paper, we explore the adaptation of policy iteration techniques to compute greatest fixpoint approximations, in order to find sufficient conditions for program termination. Restricting ourselves to affine programs and the abstract domain of template constraint matrices, we show that the abstract greatest fixpoint can be computed exactly using linear programming, and that strategies are related to the template constraint matrix used. We also present a first result on the relationships between this approach and methods which use ranking functions.

Keywords: Abstract interpretation, policy iteration, template constraint matrices, termination analysis.

1 Introduction

Abstract interpretation [5] is a powerful framework to develop program analyses. Most analyses require the computation of approximation of fixpoints on an abstract domain, either least fixpoints (*lfp*) or greatest fixpoints (*gfp*). The traditional method to compute approximations of fixpoints uses widening and narrowing operators. Widening operators are designed to get beyond the fixpoint, which makes them useful to compute overapproximations of *lfp*, or (dually) underapproximations of *gfp* [6]. However, used on state abstractions, these approximations can only be used to check safety properties. Liveness properties (and especially termination) can be proved by computing underapproximations of *lfp* or overapproximations of *gfp*.

More recently, other approaches have been developed to compute abstract fixpoints: abstract acceleration [12] and policy iteration [3,9]. Both methods

[1] Email: damien.masse@univ-brest.fr

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.com/locate/entcs

are designed in order to compute the exact abstract fixpoint for specific transfer functions. They have been used to compute reachability analyses (which involves the computation of an *lfp*), and since the abstract domains used were overapproximating domains, they were applied to prove safety properties, providing a greater precision than widenings. Using these approaches to compute greatest fixpoints would enable the discovery of sufficient conditions for program termination, and as a particular case proving termination for all inputs.

This paper describes the use of policy iteration techniques to discover sufficient termination conditions. As a first work in this direction, we restrict ourselves to affine programs and to the template constraint matrices abstract domain [14], a sub-domain of polyhedra. Policy iteration techniques were already used in this framework to approximate the set of reachable states [10], hence we need to adapt these results to greatest fixpoint computations.

We first present the relationships between approximating fixpoints and proving termination. Then we give an overview of the policy iteration approach. In section 4, we explore the extension of these approaches to the computation of an abstract backward semantics designed to prove termination properties. Finally we give a first result on the relationships between our approach and ranking function synthesis, showing that programs admitting a linear ranking function can be treated with our approach.

2 Program termination and fixpoint approximation

In this section, we recall a few results on the relationships between termination and fixpoint approximation. A program P is defined as a transition system (Σ, τ), Σ being an (infinite) set of states and $\tau \subseteq \Sigma \times \Sigma$ a transition relation. Furthermore, we consider S_0 as the set of initial states.

The trace semantics of a program P is the set of finite and infinite execution traces of P. The program is said to (definitely) terminate from S_0 if any execution trace starting from $s_0 \in S_0$ is finite. Broadly, three approaches can be used to show this property.

Variant abstraction analysis. Many methods use a kind of variant abstraction analysis [7], where one finds a mapping r from the set of reachable states (from S_0) to a well-founded set $(O, <)$, such that for any transition $\langle \sigma, \sigma' \rangle \in \tau$ we have $r(\sigma') < r(\sigma)$. Once the class of variant functions (or the variant abstraction) is fixed, the analysis can be expressed as a safety analysis.

Least fixpoint underapproximation. An alternative approach is to prove that S_0 is included in the set of states which could only terminate, that is:

$$S_0 \subseteq \operatorname{lfp} \lambda X. \widetilde{\operatorname{pre}}(X) \tag{1}$$

where $\widetilde{\operatorname{pre}}(X) = \{y \in \Sigma \mid \forall x \in \Sigma, \langle y, x \rangle \in \tau \Rightarrow x \in X\}$.

This property requires to *underapproximate* the least fixpoint. As noted

in [7], underapproximations are not much used in practice (most abstract domains, in particular numerical abstract domains, are designed to handle overapproximations). Furthermore, one cannot use a the "classical" fixpoint induction techniques (with widenings) to underapproximate least fixpoint.

Greatest fixpoint overapproximation. Similarly, we can show that S_0 is disjoint from the set of states that are potentially non-terminating, i.e.:

$$S_0 \cap \operatorname{gfp} \lambda X.\operatorname{pre}(X) = \emptyset \tag{2}$$

where $\operatorname{pre}(X) = \{y \in \Sigma \mid \exists x \in X, \langle y, x\rangle \in \tau\}$.

Proving this property can be done by *overapproximating* the greatest fixpoint. Compared to the (formally equivalent) previous approach, using overapproximations has the advantage of being compatible with most abstract domains. However, we still cannot use widenings to approximate the fixpoint.

The three approaches are related: with a ranking function, one can prove formulas (1) and (2). Reciprocally, proving formula (1) or (2) proves that a ranking function exists. In fact, some lower fixpoint induction methods (e.g. in [4, Sect. 11]) directly use some kind of ranking function. However, if the approximation is proved with other methods, it may be difficult to make the ranking function explicit. Hence, techniques which compute directly a fixpoint appear as interesting alternatives to infer termination properties.

3 Precise fixpoint approximation with policy iterations

The use of policy iterations (also called strategy iterations) to compute the least fixpoint of a self-map f in static analysis was first introduced in [3]. The principle of this techniques is to describe f as the minimum (or the maximum) of a set S of simpler maps. A *strategy* (or a *policy*) is a selection of an element of S. The least fixpoint of this element is computed. If this fixpoint is a fixpoint of f, the algorithm terminates, otherwise a new strategy is selected during the *strategy improvement* step, and the algorithm iterates.

Two different approaches have been proposed to compute least fixpoints: the first one [3,8,1] uses min-strategy iteration, approaches the least fixpoint from above, and does not guarantee to return it in the general case[2]. The second one [9,10,11] uses max-strategy iteration, approaches the least fixpoint from below, and guarantees to return the least (abstract) fixpoint.

Since our goal is to overapproximate greatest fixpoints, it seems more natural to approach them from above, hence to use a dual version of the second approach. In this paper, we mainly follow the method presented in [10] and restrict ourselves to affine programs and template constraint matrix domains. Before summarizing the method, we introduce a few notations.

[2] However, it does return the least fixpoint in the case of nonexpansive self-maps.

3.1 Notations

In the following, $X = \{x_1, \ldots, x_k\}$ denotes a tuple of variables. An *assignment* ρ on X is defined as a mapping from X to $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. When there is no ambiguity, ρ may be represented as an element of $\overline{\mathbb{R}}^k$. The order relation $\leq$ (and its strict version $<$) on $\overline{\mathbb{R}}$ is extended component-wise to $\overline{\mathbb{R}}^k$. We denote by $\vee$ and $\wedge$ the minimum and maximum operators on $\overline{\mathbb{R}}$ (and their component-wise extension to $\overline{\mathbb{R}}^k$). Hence $\overline{\mathbb{R}}^k$ has a complete lattice structure.

If ρ is an assignment on X, we denote by $[\rho]_f$ (resp. $[\rho]_\infty$, $[\rho]_{-\infty}$) the set of x in X such that $\rho(x)$ is finite (resp. equal to $+\infty$, $-\infty$).

If m is a function from $X \to \overline{\mathbb{R}}$ to $\overline{\mathbb{R}}$, $\mathrm{dom}(m)$ represents the set of assignments ρ such that $m(\rho)$ is finite, and $\mathrm{fdom}(m) = \mathrm{dom}(m) \cap \mathbb{R}^X$. The function m is said to be order-convex (resp. order-concave) iff $\mathrm{fdom}(m)$ is convex [3] and for all comparable ρ, ρ' in $\mathrm{fdom}(m)$ and $\lambda \in [0,1]$, $\lambda m(\rho) + (1-\lambda)m(\rho') \geq m(\lambda\rho + (1-\lambda)\rho')$ (resp. $\lambda m(\rho) + (1-\lambda)m(\rho') \leq m(\lambda\rho + (1-\lambda)\rho')$).

Let D be a monotonic function from $X \to \overline{\mathbb{R}}$ to $X \to \overline{\mathbb{R}}$. A *prefixpoint* (resp. *postfixpoint*) of D is an assignment ρ such that $D(\rho) \geq \rho$ (resp. $D(\rho) \leq \rho$). If ρ is a prefixpoint (resp. postfixpoint) of D, we denote by $\mathrm{lfp}_{\geq\rho} D$ (resp. $\mathrm{gfp}_{\leq\rho} D$) the least (resp. greatest) fixpoint of D greater than (resp. lower than) ρ.

Finally, an *equation system* $\mathcal{E}$ on X is a k-tuple of equations $(x_1 := e_1, \ldots, x_k := e_k)$ where e_i are expressions using the variables X. If $[\![\cdot]\!]$ represents the semantics of expressions (such that $[\![e_i]\!] \in (X \to \overline{\mathbb{R}}) \to \overline{\mathbb{R}}$), the semantics of $\mathcal{E}$ is defined as:

$$[\![\mathcal{E}]\!] : (X \to \overline{\mathbb{R}}) \to (X \to \overline{\mathbb{R}})$$
$$[\![\mathcal{E}]\!]\,\rho : x_i \mapsto [\![e_i]\!]\,\rho$$

A solution (resp. postsolution, presolution) of $\mathcal{E}$ is a fixpoint (resp. postfixpoint, prefixpoint) of $[\![\mathcal{E}]\!]$.

3.2 Computing the least solution of a system of equations

We consider an equation system $\mathcal{E}$ on X where the expressions are defined by the grammar:

$$e ::= a \mid x_i \mid e + e \mid b \cdot e \mid e \vee e \mid e \wedge e$$

where $a \in \overline{\mathbb{R}}$, $b \in \mathbb{R}^{>0}$, $\vee$ is the max operator and $\wedge$ is the min operator. The semantics of e is straightforward:

$$[\![a]\!]\,\rho = a \quad [\![e_1 + e_2]\!]\,\rho = [\![e_1]\!]\,\rho + [\![e_2]\!]\,\rho \quad [\![b \cdot e]\!]\,\rho = b\,[\![e]\!]\,\rho$$
$$[\![x_i]\!]\,\rho = \rho(x_i) \quad [\![e_1 \vee e_2]\!]\,\rho = [\![e_1]\!]\,\rho \vee [\![e_2]\!]\,\rho \quad [\![e_1 \wedge e_2]\!]\,\rho = [\![e_1]\!]\,\rho \wedge [\![e_2]\!]\,\rho$$

[3] For all ρ, ρ' in $\mathrm{fdom}(m)$ and $\lambda \in [0,1]$, $\lambda\rho + (1-\lambda)\rho \in \mathrm{fdom}(m)$.

The least solution of $\mathcal{E}$ can be computed using max strategy iteration [10]:

- a max-strategy π is a function mapping every expression $e_1 \vee e_2$ to a subexpression e_1 or e_2; applying π to $\mathcal{E}$ gives a system of *conjunctive equations* (without the $\vee$ operator) $\mathcal{E}_\pi$;
- the least solution μ_π greater than a current presolution of $\mathcal{E}_\pi$ is computed by solving two linear programs extracted from the system in linear time;
- the computation terminates if μ_π is a solution $\mathcal{E}$, otherwise a new strategy π' is selected (such that $[\![\mathcal{E}]\!](\mu_\pi) = [\![\mathcal{E}_{\pi'}]\!](\mu_\pi)$), and the computation loops.

In [10] and [11], the notions of *consistent presolution* and *feasible presolution* are defined to ensure the validity and the termination of this approach. In section 4, the dual notions will be used to compute greatest fixpoints.

3.3 Systems of rational equations with linear programs

Equations with linear programs (LPs) are defined by adding $LP_{A,b}(e,\ldots,e)$ in the grammar of expressions, with $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^n$. The semantics of $LP_{A,b}$ is defined as:

$$[\![LP_{A,b}(e_1,\ldots,e_m)]\!]\,\rho = \bigvee\{b^T x \mid x \in \mathbb{R}^n, Ax \leq ([\![e_1]\!]\,\rho,\ldots,[\![e_m]\!]\,\rho)\}$$

Rational equations with linear programs are used to express the abstract semantics of affine programs in the *template constraint matrix* domain [14].

In [10], Gawlitza and Seidl show that LP subexpressions can be handled during the resolution of the system of conjunctive equations by adding new variables and inequations. In [11], this result is generalized to *order-concave equations*, using the fact that the operator $\wedge$ and LP expressions are order-concave. Since the backward semantics of programs also use LP expressions, this result cannot be applied directly to compute overapproximations of greatest fixpoints: one would need order-convex expressions.

4 Computation of the backward semantics

4.1 Backward semantics of the program

We consider affine programs as a triple $(N, E, \mathbf{st})$ where N is a finite set of program points, $E \subseteq N \times \mathbf{Stmt} \times N$ is a finite set of transitions labeled by *statements*, and $\mathbf{st}$ is the *start program point*. A statement is a pair $(g; a)$ where g is an affine *guard* $A\mathbf{x} + b \geq 0$ on the set of program (real) variables $\mathbf{x} = (x_1,\ldots,x_n)$ and a is an affine *assignment* $\mathbf{x} := A\mathbf{x} + b$.

The *backward collecting semantics* of a statement $s = (A\mathbf{x} \leq b; \mathbf{x} :=$

$C\mathbf{x} + d$) is defined as:

$$[\![s]\!] : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$$
$$[\![s]\!](X) = \{x \in \mathbb{R}^n \mid Ax + b \geq 0 \wedge Cx + d \in X\}$$

The backward transformer pre on $N \to \wp(\mathbb{R}^n)$ is defined as $\text{pre}(X)(u) = \bigcup_{(u,s,v)\in E} [\![s]\!](v)$. Our goal is to overapproximate $\mathcal{B} = \text{gfp pre}$. We use the framework of abstract interpretation, and our approach is closely related to the abstract domain used (we want to compute exactly the abstract fixpoint).

The abstract domain (first introduced in [14]) is relative to a *template constraint matrix* $T \in \mathbb{R}^{m\times n}$. Each row of T represents a linear combination of program variables. The matrix T defines an abstraction from $\mathbb{R}^n$ to $\mathcal{T}_T = \overline{\mathbb{R}}^m$ with the Galois connection $\mathbb{R}^n \underset{\alpha_T}{\overset{\gamma_T}{\leftrightarrows}} \mathcal{T}_T$:

$$\gamma_T(\rho) = \{x \in \mathbb{R}^n \mid Tx \leq \rho\} \qquad \alpha_T(X) = \vee\{\rho | \gamma_T(\rho) \subseteq X\}$$

The functions γ_T and α_T are extended component-wise to $N \to \mathbb{R}^n$ and $N \to \mathcal{T}_T$. An element of $\gamma_T(\overline{\mathbb{R}}^m)$ is said to be *canonical*. The best abstract backward semantics of a statement s in this domain is defined as $[\![s]\!]^\sharp = \alpha_T \circ [\![s]\!] \circ \gamma_T$.

Lemma 4.1 *Let $s = (A\mathbf{x} + b \geq 0; \mathbf{x} := C\mathbf{x} + d)$ be a statement, and T a non-empty template matrix. Let ρ be an abstract value on the domain $\mathcal{T}_T$.*

Let A', b' and ρ' be defined as:

$$A' = \begin{pmatrix} -A \\ TC \end{pmatrix} \quad b' = \begin{pmatrix} -b \\ Td \end{pmatrix} \quad \rho' = \begin{pmatrix} 0 \\ \rho \end{pmatrix} \tag{3}$$

Then $[\![s]\!]^\sharp$ satisfies:

$$[\![s]\!]^\sharp_i(e) = \begin{cases} -\infty & \text{if } \{x \mid A'x + b' - \rho' \leq 0\} = \emptyset \\ \bigwedge\{(\rho' - b')^T\lambda \mid \lambda \geq 0 \wedge A'^T\lambda = T_i\} & \text{otherwise} \end{cases}$$

Remark 4.2 If $\{x \mid A'x + b' - \rho' \leq 0\} \neq \emptyset$ and $\{\lambda \mid \lambda \geq 0 \wedge A'^T\lambda = T_i\} = \emptyset$, then $[\![s]\!]^\sharp_i(\rho) = \infty$. Furthermore, $\{x \mid A'x + b' - \rho' \leq 0\} \neq \emptyset$ implies $\min\{(\rho' - b')^T\lambda \mid \lambda \geq 0 \wedge A'^T\lambda = T_i\} > -\infty$ (but the converse is not true).

Example 4.3 With only one variable x_1, $s = (0 \geq 0; x_1 = 0)$ and $T = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$, we have $\bigwedge\{(\rho' - b')^T\lambda \mid \lambda \geq 0 \wedge A'^T\lambda = T_i\} = \infty$ for all i and all $\rho = (\rho_1, \rho_2)$. However, if $\rho_1 < 0$ or $\rho_2 < 0$, then $\{x \mid A'x + b' - \rho' \leq 0\} = \emptyset$. Thus, by defining $(\rho'_1, \rho'_2) = [\![s]\!]^\sharp(\rho_1, \rho_2)$, we have $\rho'_1 = \rho'_2 = -\infty$ if $\rho_1 < 0$ or $\rho_2 < 0$, and $\rho'_1 = \rho'_2 = \infty$ otherwise.

The abstract domain for a program $(N, E, \mathbf{st})$ is $N \to \mathcal{T}_T$ [4]. With $X \in$

[4] For the sake of simplicity, we consider only one global template constraint matrix.

$N \to \mathcal{T}_T$, the abstract backward transformer $\mathrm{pre}^\sharp = \alpha_T \circ \mathrm{pre} \circ \gamma_T$ is given by:

$$(\mathrm{pre}^\sharp(X)(u))_i = \bigvee_{(u,s,v)\in E} [\![s]\!]_i^\sharp\,(X(v))$$

Lemma 4.4 *The abstract semantics* $\mathrm{gfp}\,\mathrm{pre}^\sharp$ *satisfies* $\gamma_T(\mathrm{gfp}\,\mathrm{pre}^\sharp) \supseteq \mathcal{B}$.

To compute the greatest fixpoint of $\mathrm{pre}^\sharp$, we describe the function as a system of semantic equations of the form $x_i = \mathrm{pre}_i^\sharp(x)$ when $\mathrm{pre}_i^\sharp$ appears as the maximum of (one or) several $[\![s]\!]_i^\sharp$. Following lemma 4.1, $[\![s]\!]_i^\sharp$ can be expressed as the minimum of two expressions ϕ_s (independent of i) and ψ_s^i:

- $\phi_s(\rho) = \begin{cases} \infty & \text{if } \{x \mid A'x + b' - \rho' \leq 0\} \neq \emptyset \\ -\infty & \text{if } \{x \mid A'x + b' - \rho' \leq 0\} = \emptyset \end{cases}$.
- $\psi_s^i(\rho) = \bigwedge\{(\rho' - b')^T\lambda \mid \lambda \geq 0 \wedge A'^T\lambda = T_i\}$

One can see that ϕ_s is monotonic, order-concave and order-convex (since $\mathrm{fdom}(\phi_s) = \emptyset$). Using the vertex principle of linear programming, we can express ψ_s^i as the minimum of a finite number of linear expressions:

Lemma 4.5 *There exists a finite (possibly empty) number of tuples* $(\lambda_1, \ldots, \lambda_k)$ *such that* $\lambda_j \geq 0$ *and* $A'^T\lambda_j = T_i$ *for all* j *and, for all* $\rho > -\infty$*:*

$$\psi_s^i(\rho) = \bigwedge_{j=1}^{k} (\rho' - b')\lambda_j$$

This equality is also satisfied when some components of ρ are equal to $+\infty$. In this case, the matching component of λ must be equal to 0.

The number of linear expressions can be exponential in the number of variables. Rather than computing all of them, we plan to lazily compute only the relevant expressions during the selection of the strategy (see remark 4.9).

Example 4.6 With two variables x and y, $s = (x - y \leq 10; \{x := -2y, y := x + 3\})$ and the octagon template matrix [13], $[\![s]\!]^\sharp$ is represented Figure 1. This example shows that the number of $\wedge$ operators is related to the template domain, both in order to deal with the potential non-canonicity of e and to ensure the canonicity of $[\![s]\!]\,(e)$. For example, if the initial assignment is canonical, we have directly $C_{-y} \leq C_{-x-y} + C_x$ and $C_{-y} \leq C_{x-y} + C_{-x}$, hence the equation of C_{-x} is equivalent to $C_{-x} := \phi \wedge C_{-y} + 3$.

From the previous lemma, we deduce:

Proposition 4.7 *The backward abstract semantics of an affine program* $(N, E, \mathbf{st})$ *in a template matrix domain can be expressed as the greatest fixpoint of a system of equations of the form:*

$$x := U_1 \vee U_2 \vee \ldots \vee U_k \quad \textit{with } U_i := \phi_i \wedge u_i^1 \wedge \ldots \wedge u_i^l$$

$$\phi = alltrue(\{C_x + C_{-x} \geq 0,\ C_y + C_{-y} \geq 0,\ C_{x+y} + C_{-x-y} \geq 0,\ C_{x-y} + C_{-x+y} \geq 0,\ C_x + C_y + C_{-x-y} \geq 0,\ C_{-x} + C_{-y} + C_{x+y} \geq 0,\ C_x + C_{-y} + C_{-x+y} \geq 0,\ C_{-x} + C_y + C_{x-y} \geq 0,\ 2C_x + C_{-x-y} + C_{-x+y} \geq 0,\ 2C_{-x} + C_{x+y} + C_{x-y} \geq 0,\ 2C_y + C_{-x-y} + C_{x-y} \geq 0,\ 2C_{-y} + C_{x+y} + C_{-x+y} \geq 0,\ 2C_{-y} + C_{-x} + 26 \geq 0,\ C_x + C_{-x-y} + 26 \geq 0,\ 3C_{-x} + 2C_{x-y} + 26 \geq 0,\ C_{-y} + C_{-x-y} + 26 \geq 0,\ 3C_{-y} + 2C_{-x+y} + 26 \geq 0,\ C_{-x-y} + C_{x-y} + 52 \geq 0\}).$$

$$\begin{aligned}
C_x &:= \phi \wedge 10 + C_x/2 \wedge 23 + C_{-x-y} \wedge (17 + 2C_{-x+y})/3 \\
&\quad \wedge C_y - 3 \wedge C_{x+y} + C_{-x} - 3 \wedge C_{-x+y} + C_x - 3 \\
C_{-x} &:= \phi \wedge C_{-y} + 3 \wedge C_{-x-y} + C_x + 3 \wedge C_{x-y} + C_{-x} + 3 \\
C_y &:= \phi \wedge C_{-x}/2 \wedge (C_{-x-y} + C_y)/2 \wedge (C_{-x+y} + C_{-y})/2 \\
C_{-y} &:= \phi \wedge C_x/2 \wedge (C_{x+y} + C_{-y})/2 \wedge (C_{x-y} + C_y)/2 \wedge 13 + C_{-y} \wedge (13 + C_{x-y})/3 \\
C_{x+y} &:= \phi \wedge C_{-x}/2 + C_y - 3 \wedge 3C_{-x}/2 + C_{x+y} - 3 \wedge (3C_y + C_{-x-y})/2 - 3 \\
&\quad \wedge (C_y + C_{-x+y})/2 - 3 \wedge (C_{x+y} + 3C_{-x+y})/4 - 3 \wedge 10 + C_{-x} \wedge 36 + 2C_{-x-y} \\
&\quad \wedge (4 + 2C_{-x+y})/3 \\
C_{-x-y} &:= \phi \wedge C_x/2 + C_{-y} + 3 \wedge 3C_x/2 + C_{-x-y} + 3 \wedge (3C_{-y} + C_{x+y})/2 + 3 \\
&\quad \wedge (C_{-y} + C_{x-y})/2 + 3 \wedge (C_{-x-y} + 3C_{x-y})/4 + 3 \wedge 16 + 2C_{-y} \\
C_{x-y} &:= \phi \wedge 10 \wedge C_x/2 + C_y - 3 \wedge 3C_x/2 + C_{-x+y} - 3 \wedge C_{-x}/2 + C_{x+y} + 3 \\
&\quad \wedge (C_y + C_{x+y})/2 - 3 \wedge (3C_y + C_{x-y})/2 - 3 \\
C_{-x+y} &:= \phi \wedge C_{-x}/2 + C_{-y} + 3 \wedge 3C_{-x}/2 + C_{x-y} + 3 \wedge C_{-x}/2 + C_{x+y} + 3 \\
&\quad \wedge (C_{-y} + C_{-x-y})/2 + 3 \wedge (3C_{-y} + C_{-x+y})/2 + 3
\end{aligned}$$

Fig. 1. Backward semantics of the transition $s = (x - y \leq 10; \{x := -2y, y := x + 3\})$ in the octagon domain, described as a system of equations. Each variable C_{exp} represents the maximum of exp in the abstract element. We denote by *alltrue* the function which maps a set of constraints to ∞ if all constraints are satisfiable, and $-\infty$ otherwise.

where ϕ_i is a monotonic function whose image is included in $\{-\infty, +\infty\}$ and u_i^j are linear expressions.

Remark 4.8 If an overapproximation of the reachable states has been computed, it can be included in the system of equations (if the abstract forward analysis returns $x = a$, the equation becomes $x := a \wedge (U_1 \vee \ldots \vee U_k)$). This combination increases the precision of the backward analysis [6].

Remark 4.9 Since the explicit computation of the system is too costly, we express each U_i as $\phi_i \wedge \psi^i$ where ψ^i is a linear program. During the strategy selection phase, an optimal u_i^j is constructed by solving the linear program with the current affectation ρ: if there is an optimal solution λ, then $\lambda(\rho' - b')$ is used (if it is an improvement compared with the current strategy), otherwise the strategy returns $+\infty$. The number of basic feasible solutions may be high, but most are related to the canonicity of the abstract elements, so we can expect the number of selected strategies to remain acceptable.

Similarly, we do not expect to compute explicitly ϕ_i as a set of constraints on ρ. Rather, we check the feasibility of the domain at each strategy iteration.

4.2 Solving the system of equations

Following the policy iteration principle, we consider a strategy associating each expression $\phi_i \wedge u_i^1 \wedge \ldots \wedge u_i^l$ (or rather $\phi_i \wedge \psi^i$) to either ϕ_i or a linear

expression u_i^k. If ρ is the current assignment, the strategy π_ρ must satisfy:

$$\pi_\rho(\phi_i \wedge u_i^1 \wedge \ldots \wedge u_i^l) = \min(\phi_i(\rho), u_i^1(\rho), \ldots, u_i^l(\rho)) \tag{4}$$

Since the image of ϕ_i is included in $\{-\infty, +\infty\}$, we can ensure that $\pi_\rho(\phi_i \wedge \ldots) = \phi_i$ only when $\phi_i(\rho) = -\infty$. Furthermore, since ϕ_i is monotonic and we compute a decreasing sequence, once $\phi_i(\rho) = -\infty$ the whole expression can be replaced by $-\infty$. Thus, the application of π_ρ gives a system of equations of the form $(x := u_1 \vee \ldots \vee u_k)$ where each u_i is either $-\infty$ (which can be ignored) or a linear expression. This system is a system of disjunctive equations.

Given a postsolution ρ of a disjunctive system $\mathcal{E}$, we want to compute $\mathrm{gfp}_{\leq\rho} [\![\mathcal{E}]\!]$. Since our approach is exactly the dual of the method proposed in [10], we just give a definition of consistency and the final theorem here.

Definition 4.10 Given a disjunctive system $\mathcal{E}$, a finite solution ν of $\mathcal{E}$ is said to be *feasible* iff there exists $\rho > \nu$ such that $[\![\mathcal{E}]\!](\rho) < \rho$. A finite postsolution ρ is *feasible* iff $\mathrm{gfp}_{\leq\rho} [\![\mathcal{E}]\!]$ is finite and feasible. A disjunctive system is *feasible* iff it admits a feasible solution.

Definition 4.11 Given a disjunctive system $\mathcal{E}$, a postsolution ρ is said to be *consistent* iff the following conditions are satisfied:

- $[\![\mathrm{exp}]\!]\,\rho = \infty$ implies $\mathrm{exp} = \infty$ for every expression exp occurring in $\mathcal{E}$;
- with $\nu = [\mathrm{gfp}_{\leq\rho} [\![\mathcal{E}]\!]]_f$, the system $\mathcal{E}'$ on ν_f defined by replacing in the equations of $\mathcal{E}$ any variable $x \in \nu_\infty$ by ∞ and $x \in \nu_{-\infty}$ by $-\infty$ is feasible, and $\rho_{|\nu_f}$ is a feasible postsolution of $\mathcal{E}'$.

Theorem 4.12 ([10, Thm 3, dual]) *Given a consistent postsolution ρ of a disjunctive system $\mathcal{E}$, $\mathrm{gfp}_{\leq\rho} [\![\mathcal{E}]\!]$ can be computed by solving two LPs extractable from $\mathcal{E}$ in linear time.*

4.3 Strategy improvement

We still need to prove that the strategy improvement operator preserves consistency. First we rewrite all equations $x := U$ as $x := +\infty \wedge U$. The initial strategy π_∞ associates each equation to $+\infty$. In the associated system, $(x_i \mapsto +\infty)$ is consistent. Consistency is preserved under three conditions:

Lemma 4.13 *Let π be a strategy, ρ an assignment and π' an improved strategy satisfying:*

(i) *for each maximum of order-convex expressions $\wedge U$, if $\pi(\wedge U)(\rho) = \pi'(\wedge U)(\rho)$, then $\pi(U) = \pi'(U)$;*

(ii) *if $\pi(\wedge U) \neq \pi'(\wedge U)$, then $\pi'(\wedge U)(\rho) < \pi(\wedge U)(\rho)$;*

(iii) *if $\pi'(\wedge U)(\rho) < \infty$, then for all $\rho' \geq \rho$ with $[\rho']_f = [\rho]_f$, $\pi'(\wedge U)(\rho) < \infty$.*

Then any consistent solution of $\pi(\mathcal{E})$ is a consistent postsolution of $\pi'(\mathcal{E})$.

#	Current strategy	Fixpoint
1	$C_x := +\infty$, $C_{-x} := +\infty$, $C_y := +\infty$, $C_{-y} := +\infty$, $C_{x+y} := +\infty$, $C_{-x-y} := +\infty$, $C_{x-y} := 10$, $C_{-x+y} := +\infty$	$x - y \leq 10$
2	$C_x := +\infty$, $C_{-x} := +\infty$, $C_y := +\infty$, $C_{-y} := (13+C_{x-y})/3$, $C_{x+y} := +\infty$, $C_{-x-y} := +\infty$, $C_{x-y} := 10$, $C_{-x+y} := +\infty$	$x - y \leq 10$, $-23/3 \leq y$
3	$C_x := +\infty$, $C_{-x} := 3 + C_{-y}$, $C_y := +\infty$, $C_{-y} := (13 + C_{x-y})/3$, $C_{x+y} := +\infty$, $C_{-x-y} := 3 + (C_{x-y} + C_{-y})/2$, $C_{x-y} := 10$, $C_{-x+y} := +\infty$	$-32/3 \leq x$, $x - y \leq 10$, $-23/3 \leq y$, $-71/6 \leq x + y$
4	$C_x := 10 + C_{-x}/2$, $C_{-x} := 3 + C_{-y}$, $C_y := C_{-x}/2$, $C_{-y} := (13 + C_{x-y})/3$, $C_{x+y} := 10 + C_{-x}$, $C_{-x-y} := 3 + (C_{x-y} + C_{-y})/2$, $C_{x-y} := 10$, $C_{-x+y} := 3 + (C_{-x-y} + C_{-y})/2$	$-32/3 \leq x \leq 46/3$, $-51/4 \leq x - y \leq 10$, $-23/3 \leq y \leq 16/3$, $-71/6 \leq x + y \leq 62/3$
5	$C_x := -3 + C_y$, $C_{-x} := 3 + C_{-y}$, $C_y := C_{-x}/2$, $C_{-y} := (13 + C_{x-y})/3$, $C_{x+y} := -3 + (C_{-x+y} + C_y)/2$, $C_{-x-y} := 3 + (C_{x-y} + C_{-y})/2$, $C_{x-y} := 10$, $C_{-x+y} := 3 + (C_{-x-y} + C_{-y})/2$	$-32/3 \leq x \leq 7/3$, $-51/4 \leq x - y \leq 10$, $-23/3 \leq y \leq 16/3$, $-71/6 \leq x + y \leq 145/24$
6	$C_x := -3+C_y$, $C_{-x} := 3+C_{-y}$, $C_y := C_{-x}/2$, $C_{-y} := C_x/2$, $C_{x+y} := -3+(C_{-x+y}+C_y)/2$, $C_{-x-y} := 3+(C_{x-y}+C_{-y})/2$, $C_{x-y} := -3+(C_{x+y}+C_y)/2$, $C_{-x+y} := 3+(C_{-x-y}+C_{-y})/2$	$x = -2$, $x - y = -3$, $y = 1$, $x + y = -1$

Fig. 2. Computation of the abstract semantics for a single state program with one transition $(x - y \leq 10; \{x := -2y, y := x + 3\})$ in the octagon domain. The initial strategy $(+\infty)$ is omitted.

Conditions (i) and (ii) are consequences of the principle of strategy improvement. The third one is satisfied because we use linear expressions. The termination of the computation is guaranteed by the finite number of strategies. As mentioned in remark 4.9, we expect the number of iterations to remain low, but more experiments are needed to validate this hypothesis.

Finally, we can state the general result on whole programs:

Theorem 4.14 *Given an affine program* $(N, E, \mathbf{st})$ *and a template matrix* T*, the algorithm terminates and returns the abstract semantics* $\mathrm{gfp}\ \mathrm{pre}^\sharp$*.*

Example 4.15 We consider a program (N, E, i) with only one program point $N = \{i\}$ and $E = \{(i, s, i)\}$ with s defined as in example 4.6. Figure 2 gives the sequences of strategies (as systems of equations) and the fixpoints (as constraints on x and y). The set of non-terminating states is included in each fixpoint (in this example, the last fixpoint is exactly the set of non-terminating states). Thus, from any initial state except $(x = -2; y = 1)$, the program terminates.

5 Relationships with variant analysis

To compare our method with variant analysis, we search a correspondence between the provability of termination with policy iteration and the kind of ranking functions which can be used to prove the termination of the program.

When the program is given, the result of the policy iteration-based analysis depends only on the abstract domain, since it computes exactly $\mathrm{gfp}\ \mathrm{pre}^\sharp$ where $\mathrm{pre}^\sharp = \alpha \circ \mathrm{pre} \circ \gamma$. First, we can see that a ranking function can be constructed

from the iteration sequence of $\text{pre}^\sharp$.

Lemma 5.1 *Let $\Sigma = N \to \wp(\mathbb{R}^n)$ be the set of states of the program, and τ the transition relation between elements of Σ. Let (W^i) be the iteration sequence of $\text{pre}^\sharp$ starting from $(x \mapsto +\infty)$ and O an ordinal such that $W^{O-1} = \text{gfp } \text{pre}^\sharp$. Then the function v defined from $Z = \Sigma \setminus \gamma_T(\text{gfp } \text{pre}^\sharp)$ to O as:*

$$v(\sigma) = \min\{i \,|\, \sigma \notin \gamma_T(W^i)\}$$

is a ranking function on (Z, τ).

Furthermore, for all $n \in O$, $v^{-1}(\{n' \in O \,|\, n' \geq n\}) \cup \gamma_T(\text{gfp } \text{pre}^\sharp)$ is canonical. This property can be reversed to deduce a condition on $\text{gfp } \text{pre}^\sharp$ from the existence of a ranking function:

Proposition 5.2 *Let $Z \subseteq \Sigma$ and $Y = \Sigma \setminus Z$. Then $\gamma_T(\text{gfp}\, pre^\sharp) \subseteq Y$ if and only if there exists a function v from Z to an ordinal O such that:*

$$\forall s \in Z, \forall s' \in \Sigma, \langle s, s' \rangle \in \tau \Rightarrow (s' \in Z \wedge v(s) > v(s'))$$

$$\forall n \in O, v^{-1}(\{n' \mid n' \geq n\}) \cup Y \in \gamma_T(N \to \mathcal{T}_T)$$

Hence, the abstract semantics would prove the termination for all inputs (i.e. $Z = \Sigma$) if there exists a ranking function v for which the successive preimages $v^{-1}(O)$, $v^{-1}(O \setminus \{0\})$, $v^{-1}(O \setminus \{0, 1\})$, ... for every program point are of the form $TX \leq B$.

Corollary 5.3 *If an affine program can be proved to terminate with a linear ranking function $X \mapsto RX$, our method proves the termination of the program if $-R$ is a row of T.*

However, the program given in example 4.15 does not admit a linear ranking function. Thus our method is not limited to linear ranking functions, and a global characterization of the ranking functions remains to be stated.

6 Conclusion and future work

This paper is intended to show how policy iteration techniques can be applied to termination analysis. We performed only a few experiments, and more comparisons with related work needs to be done. Recently, Bozga et al. [2] presented several results on the decision of conditional termination. Their framework is more restrictive (as they restrict themselves to integer variables), but may give more precise results as it is not limited by the precision of the abstract domain.

To improve the analysis, we can extend previous work on policy iterations for other abstract domains (e.g. quadratic zones [11]). Although we dealt only with affine programs, we can also add linearization and non-determinism

to extend the framework. Finally, an interesting feature of greatest fixpoint overapproximation, compared with least fixpoint overapproximation, is that the abstract domain can be refined during the computation. To refine the precision of the analysis, we can add new template constraints at any time, using for example decreasing iterations on the domain of polyhedra.

Acknowledgments. I wish to thank David Monniaux, as well a the anonymous referees, for their comments and suggestions.

References

[1] Adjé, A., S. Gaubert and E. Goubault, *Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis*, in: A. D. Gordon, editor, *ESOP*, Lecture Notes in Computer Science **6012** (2010), pp. 23–42.

[2] Bozga, M., R. Iosif and F. Konecný, *Deciding conditional termination*, in: *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS*, Lecture Notes in Computer Science **7214** (2012), pp. 252–266.

[3] Costan, A., S. Gaubert, E. Goubault, M. Martel and S. Putot, *A policy iteration algorithm for computing fixed points in static analysis of programs*, in: K. Etessami and S. K. Rajamani, editors, *CAV*, Lecture Notes in Computer Science **3576** (2005), pp. 462–475.

[4] Cousot, P., *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation*, Theoretical Computer Science **277** (2002), pp. 47–103.

[5] Cousot, P. and R. Cousot, *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*, in: *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (1977), pp. 238–252.

[6] Cousot, P. and R. Cousot, *Abstract interpretation frameworks*, Journal of Logic and Computation **2** (1992), pp. 511–547.

[7] Cousot, P. and R. Cousot, *An abstract interpretation framework for termination*, in: *Conference Record of the 39$^{\text{th}}$ Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (2012), pp. 245–258.

[8] De Nicola, R., editor, "Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007, Proceedings," Lecture Notes in Computer Science **4421**, Springer, 2007.

[9] Gawlitza, T. and H. Seidl, *Precise fixpoint computation through strategy iteration*, in: De Nicola [8], pp. 300–315.

[10] Gawlitza, T. and H. Seidl, *Precise relational invariants through strategy iteration*, in: J. Duparc and T. A. Henzinger, editors, *CSL*, Lecture Notes in Computer Science **4646** (2007), pp. 23–40.

[11] Gawlitza, T. and H. Seidl, *Computing relaxed abstract semantics w.r.t. quadratic zones precisely*, in: R. Cousot and M. Martel, editors, *SAS*, Lecture Notes in Computer Science **6337** (2010), pp. 271–286.

[12] Gonnord, L. and N. Halbwachs, *Combining widening and acceleration in linear relation analysis*, in: *13th International Static Analysis Symposium, SAS'06*, Seoul, Korea, 2006.

[13] Miné, A., *The octagon abstract domain*, Higher-Order and Symbolic Computation **19** (2006), pp. 31–100, http://www.di.ens.fr/~mine/publi/article-mine-HOSC06.pdf.

[14] Sankaranarayanan, S., H. B. Sipma and Z. Manna, *Scalable analysis of linear systems using mathematical programming*, in: R. Cousot, editor, *VMCAI*, Lecture Notes in Computer Science **3385** (2005), pp. 25–41.